A Contingency Perspective on External Component Reuse and Software Project Success

Full paper

Anisa Stefi
Ludwig-Maximilians-Universität München
stefi@bwl.lmu.de

Karl R. Lang
Baruch College
Karl.Lang@baruch.cuny.edu

Thomas Hess
Ludwig-Maximilians-Universität München
thess@bwl.lmu.de

Abstract

Software reuse can lower costs and increase the flexibility of the software development process. Despite a large body of research focused on technical factors, there is still limited research on how companies reuse exiting components. In this study, we analyzed the reuse of external software components by taking a contingency approach. Using a survey of IT managers in the software industry, we empirically found that the use of external software components in an organization leads to better outcomes of the software development process. Among large companies, organic organizations adopt external reuse strategies more aggressively than mechanistic organizations. Architecture modularity is a significant driver of software reuse strategies. Finally, our findings suggest that some organizations may view external reuse as a long-term strategy that allows them to organize and deploy resources to achieve efficiency. External software reuse can thus be seen as an effective organization strategy to improve software project success.

Keywords

Software Reuse, Component Reuse, Contingency Theory

Introduction

Information systems (IS) drive organizations and play a major role in every area of our lives. Being able to develop new software swiftly in response to new market demands or changed customer preferences, and to keep it up to date in a constantly changing technology environment, is a critical organizational capability that presents a source of competitive advantage for many organizations. One way to deal with such complexity when developing software systems is by taking advantage of reusing existing knowledge. One specific knowledge reuse approach in software development that has gained considerable interest in industry practice is the reuse of software components when developing new products. In this study, we define a software component as units of independent production, acquisition and deployment that provides an interface, which allows its functionalities to be integrated into other software products (Brereton and Budgen 2000).

The IS literature on software reuse has mostly focused on systematic internal reuse, that is, how organizations reuse their own developed assets across different projects (Frakes and Isoda 1994; Sherif et al. 2006). This kind of reuse requires up-front investment, resources, and knowledge to implement a formalized and standardized reuse process, which is expensive and challenging to do, especially for small organizations with few software projects or products that do not have a high degree of similarity (Buxmann et al. 2012). While the term reuse generally refers to the use of already developed software artifacts that were implemented previously within the organization, its original initial definition described software systems composed of already existing software components, without restricting only to components developed within the organizations (see McIlroy 1969). Ravichandran and Rothenberger

(2003) are among the few authors to classify software components by considering their origin (developed in-house vs. purchased from the market). The reuse of external components that have not been developed within the organization has rarely been investigated. While markets for software components have been suggested, at a theoretical and conceptual level, in the literature as a market mechanism to facilitate external software reuse in organizations (Ravichandran and Rothenberger 2003), there has been little empirical evidence that external software reuse could be a viable organizational software development strategy. However, in one recent study, Shang et al. (2012) did find some experimental support for external software reuse approaches. Our paper follows up on this earlier work and offers to the best of our knowledge the first empirical study that investigates actual business practice in applying external software reuse strategies.

An example of the importance of external reuse is observed in the open source community, where reuse also includes the use of existing frameworks, libraries, or code from other projects. This kind of reuse within the open source community has been shown to improve product quality and time-to-market (Sojer and Henkel 2010). In this paper, to distinguish between reuse of artifacts within an organization, which has been the dominant research stream, we refer to the use of external software components as external reuse. With the increasing availability of software artifacts and the success of software ecosystems (which rely on external knowledge), it is crucial to analyze the effect of external software components within software developing organizations. It is important to distinguish that in the case of the ecosystems, the organizations developing external complementary innovation do not necessary buy or sell to other organizations within the ecosystem (Gawer and Cusumano 2014). It must be pointed out that platform providers, assuming that they gain the rights from the complementors, can integrate such external functionality to the core of the platform, adopting an external software reuse strategy. There is only a nascent literature on these external reuse strategies. Stefi and Hess (2015) for example, view external components usage as a make-or-buy decision at a project level.

In this study, we specifically consider organizations that have in-house software development capabilities (make capabilities) and also choose to reuse software components that were developed elsewhere. While the use of external components in software projects has been investigated to some extent in the open source research (Ayala et al. 2009), organizational aspects have rarely been investigated. To address this gap, in this work we focus on the contingency approach and examine contingency factors that contribute to the adoption of external software reuse strategies. Through a contingency lens, we assume that there is no best way to manage an organization, and we investigate the contingency factors that could drive external reuse. Moreover, we analyze how external reuse influences software projects within organizations. Therefore, we examine the following two specific research questions:

(1) To what degree do external reuse strategies improve software project success in organizations with software development capabilities?

(2) How do organizational characteristics influence the reuse of external software components in organizations with software development capabilities?

Addressing these questions is important, as it helps to better understand the implementation of software reuse strategies that shape organizations. The reminder of the paper is structured as follows. First, we introduce the theoretical background on software reuse. Second, we derive the hypotheses and in the subsequent section we present the data collection procedure in detail. The following section presents the data analysis. Finally, we conclude by discussing the findings, as well as theoretical and practical implications.

Theoretical Background

Software Reuse

The idea of software reuse was introduced in 1969 by McIlroy, who envisioned software systems composed of already existing software components, similar to other mature engineering disciplines (McIlroy 1969). The idea of software reuse promises to reduce the time-to-market, to lower the software development costs, and to increase software quality (Frakes and Kang 2005). Research on software reuse has been focusing on the role of systematic software reuse in software development Behavioral factors such as lack of top management support and misalignment of short-term and long-term goals were found

to be the main barriers of successful adoption of software reuse strategies (Frakes and Isoda 1994). Shang et al. (2012) present an economic market experiment where software developers could trade reusable components, finding that the developers in their lab experiment did not adopt aggressive external reuse strategies, despite possible economic benefits from doing so. In this study we offer a fresh theoretical perspective on organizational adoption of reuse strategies by looking specifically at contingency factors within organizations.

Contingency Theory

The underlying assumption about contingency theory (CT) is that there is no one best way to organize a successful organization. It was developed in the 1950s (Burns and Stalker 1961; Woodward 1958) and sought to reflect differences between different organizations. There are number of studies in IS that adopt the contingency approach. Benlian and Hess (2007) adopted the CT to analyze the allocation of media content in publishing companies. Wong et al. (2011) looked at how contingency variables affect information integration in supply chain management. One of the main criticisms of CT is that its concepts are not clearly defined; this is reflected in this research stream. Therefore, the goal of most of the studies is to analyze the relationships among set of variables. Some of the variables found in the MIS research are strategy, structure, size, environment, task and individual characteristics (Weill and Olson 1989). Analogously to the information integration literature (see Wong et al. 2011), we argue that integrating external software functionalities into a product is a strategic action that benefits software projects, since it provides more flexibility. However, organizations should not neglect the role of situational conditions. Situational conditions are produced by an organization's internal operating or characteristics of the environment. Thus, the contingency approach provides an appropriate lens to analyze external software reuse in different organizations. In this work, we ground the contingency factors on the software reuse literature. Therefore, in addition to organization size and organization age, we also include the following contingency factors. First, we consider the structure of the organization, which can facilitate or hinder the changing IT processes (Baumann 2009). Second, we include the current strategic focus of the company that is its long-term-orientation. We include this variable, because a long-term view would provide the organization and the IT department with more resources. As pointed out in the related work on software reuse, the short-term goals of a project might differ from the organizational goals. We also consider the architecture modularization of the developed software product. Although, architecture modularity is not an organizational factor, we consider it here, since it can reflect the investment of the organization in a well-designed product. Thus, it can be considered as facilitating resources from the organizational perspective.

Research Model and Hypotheses

In this section, we will further elaborate our hypotheses and integrate them into our research model.

Decentralized Organizational Structure

Organizational context plays an important role in technology assimilation. When acquiring an external software component, the organization should consider the technology adopters' expertise, which in our particular context are software developers. Different organizations have different ways to achieve this. Burns and Stalker (1961) describe two sides of the organization structure spectrum: the mechanistic and the organic organization. Mechanistic organizations have clearly defined centralized, hierarchical structures and corresponding governance structures in place, while organic organizations are more flexible, decentralized organizations that implement informal controls and encourage open communication. In the case of software development, both developers and management need to combine their expertise in order to make better decisions, because software development requires both technical and business know-how. Therefore, in a centralized organization structure, developers have additional costs since they need to overcome the structural inertia. For instance, even when they are aware of an adequate external solution, individual developers might be reluctant to inform the management in a mechanical organization. In an organic organization, on the other hand, owing to the flat structure and communication style, we would expect a higher level of external software component reuse. Thus, organization structure, which describes "the extent to which organizations are structured in an organic versus mechanistic manner" (Covin and Slevin 1988, p.225) will influence the extent of external reuse.

H_1: A decentralized organization structure is positively associated with the reuse level of external software components within software product development.

Long-term Orientation (LTO)

Time is an important aspect in decision-making. Intertemporal choices are defined as "decisions in which the timing of costs and benefits are spread out over time" (Loewenstein and Thaler 1989, p. 181). From management's perspective, intertemporal choices consider decisions which require balancing short-term gains versus long-term goals (Laverty 1996). In the innovation literature, the LTO construct is often used to measure the characteristic of an organization that deliberately plans for requirements in the long-term (Herrmann et al. 2007). Thus in this literature stream, researchers acknowledge long-term orientation as an important attribute for the success of innovations (Herrmann et al. 2007). "Strategic decisions with a short-term orientation emphasize efficiency, whereas decisions with a long-term orientation emphasize effectiveness" (Wang and Bansal 2012, p. 1139). In the context of software product development, organizational LTO will result in a lower use of external software components. In the long-term, organization might want to design and develop components themselves as they prefer to be less dependent on other companies and to have more control over future requirements. Thus, organization with a LTO perspective will adopt more in-house development which might require additional technical and business know-how. On the other side organization with lower LTO, will be reusing existing software components in order to meet current time-to-market needs. Companies with a long-term orientation will therefore limit external components reuse. Thus, we hypothesize the following:

H_2: An organization' long-term orientation is negatively associated with the reuse level of external software components, within software product development.

IT Architecture Modularity

The concept of modularity has attracted the attention of a number of scholars in IS. A complex system "is said to exhibit modularity in design if its parts can be designed independently but will work together to support the whole" (Baldwin and Clark 2006, p. 1117). Similarly, Tiwana (2008) refers to modularity as "the degree of intentional decoupling among constituent subsystems" (Tiwana 2008 p.771). Modularity was also found to lower the need for intensive inter-firm interactions during the development process (Tiwana 2008). Research on IS outsourcing has shown that there is a connection between modularity and the outsourcing of modules, but the direction of this connection is not yet clear (Fixson et al. 2005). In this study, architecture modularity refers to the degree to which the software product being developed could be decoupled. Two distinct features of modularity are loose coupling and standardization (Tiwana and Konsynski 2010). Loose coupling is defined as the design feature of an architecture, which allows the modification of one component without affecting the behaviors of other components within the system (Tiwana and Konsynski 2010). Standardization refers to the "degree to which organization wide standards and policies pre-specify how applications in an organization's IT portfolio connect and interoperate with each other" (Tiwana and Konsynski 2010 p.3). For example, a standard could be the application programming interfaces (APIs) that describe how software applications could operate with other applications. "Standardization of software development is a complex construct, because a number of its aspects could be specified above the project level" (Nidumolu 1996, p.137). The use of standards can lead to the quick replacement of components, suggesting that the more standards are established within an organization, the easier it is to integrate components from outside. Thus, to achieve a certain degree of modularity, both loose coupling and standardization could be employed. Therefore, we adopt loose coupling and standardization as formative dimensions of modularity. We argue that a high product architecture modularity allows an easier integration of external software components. A modular architecture will also not disrupt the work of modules developed in-house. Therefore, we state the following:

H_3: The level of architecture modularity of the software product developed within the organization is positively associated with the reuse level of external software.

Success of Software Development Projects

The adoption of software reuse is linked to a number of benefits in the literature. These include a faster time-to-market, as developers do not need to reinvent the wheel, but can rely on existing solutions, and also promises to deliver high-quality software. This is even more so in the case of external software reuse, where the software components' quality is guaranteed by vendors or tested by the open source community or other customers. Additionally, component prices in the market are expected to fall over time, and thus benefit the organizations that adopt such reuse strategies. Thus, external software reuse can improve the flexibility of software development process. Therefore, we can state our hypothesis:

H_4: The reuse of external software components is positively associated with the success of software projects within the organization.

Research Model

Additionally to the above factors, we also consider organization size and organization age. Both of these organizational attributes are expected to negatively influence the level of external reuse. Big organizations tend to have multiple projects and products and seek to reuse software components that are developed internally. Similarly, older organizations, through experience, will rely more on the internal components implemented over the years. Thus, organizations would first adapt solutions within their organization and would then consider using external software components. Figure 1 below depicts our research model.

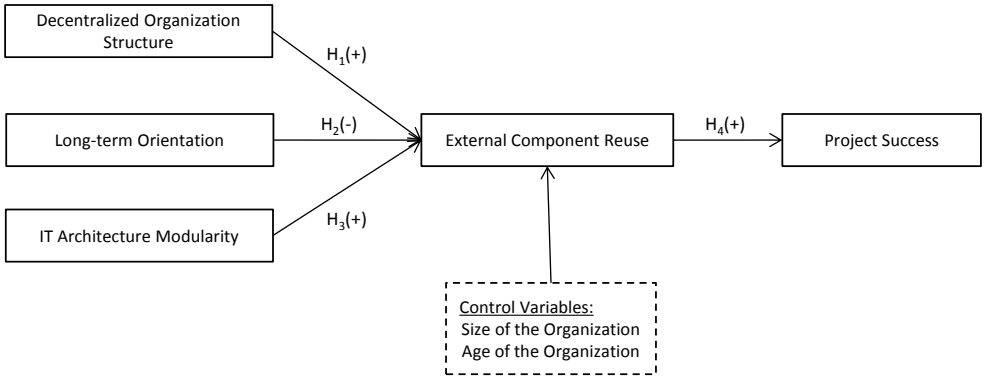

Figure 1: Research Model

Research Methodology

Data Collection and Sample

In order to test our hypotheses, we developed an online questionnaire targeting IT managers concerned with software development. At the start of the survey, we provided the definition and some examples of software components. After several pretests with professional experts and researchers, we distributed the survey to a sample of IT managers, which was randomly selected from InfoUSA and Amadeus, firm databases. The sample included only representative of the software industry identified with the Standard Industrial Classification (SIC) code 737 (computer programming, data processing, and other computer related services). After removing incomplete answers, 139 responses were considered for the analysis. About, 83.5% of our respondents were working in a management position, mostly upper management (e.g. the CEO). The sample included 63.3% software vendors, while the rest indicated that were involved in healthcare, manufacturing, finance, consulting, etc. Of the companies, approximately 72.7% had revenues of less two $2 million and 10.8% had revenues between $2 million to 10 million, and 10.1% had revenues between $11 million to $50 million while the rest of the companies had revenues between $51 million to $999 million. About 46% of the companies spend 5% to 20% of their budgets on the procurement of external software components.

Operationalization of Constructs and Instrument Validation

We operationalized all the measurements using a multi-item, 7-point Likert scale.

Constructs / Based on	**Questionnaire Items**	
Success (SUC) / Pavlou and El Sawy (2006)	To what extent do you agree or disagree with the following statements.	
	SUC01	Our software development projects/products usually finish within budget.
	SUC02	Our software development projects/products usually finish on schedule.
	SUC03	Productivity of our development staff is high compared to other similar organizations in similar environment.
External Component Reuse (ECR)/ Verwaal et al. (2008)	Please state to what extent do you agree or disagree with the following statements regarding the usage of external software components in your organization. *(Please do not count functionalities from libraries that are part of your development language)*	
	ECR01	My organization engages frequently in using external software components.
	ECR02	My organization often uses different external software components.
	ECR03	Top management has a positive attitude towards using external software components.
Decentralized Organization Structure (OS) /Covin and Slevin (1988)	The operating philosophy of the top management is:	
	OS01	Tight formal control of most operations by means of sophisticated control/Loose, informal control; heavy dependence on informal relations and norm of co-operation for getting work done
	OS02	Strong emphasis on always getting personnel to follow the formally laid down procedures/Strong emphasis on getting things done even if this means disregarding formal procedures
	OS03	A strong emphasis on holding fast to true and tried management principles despite any changes in business conditions/A strong emphasis on adapting freely to changing circumstances without too much concern for past practice
Long-term Orientation (LTO)/ Herrmann et al. (2007)	Please state, to what extent do you agree or disagree with the following statements.	
	LTO01	Quarterly profits are not the primary financial objective.
	LTO02	New software development projects do not always require rapid payback.
	LTO03	A positive margin in the long-term is most important for new software projects.
Loose Coupling (LC)/ Tiwana and Konsynski (2010)	Please estimate the extent to which the following characteristics describe the architecture of the software developed in your company.	
	LC01	Loosely coupled
	LC02	Highly modular
Standardization (ST)/ Tiwana and Konsynski (2010)	Please estimate the extent to which the following aspects of IT are established in at the enterprise wide level*:	
	ST01	IT policies
	ST02	IT architecture
Notes: The scale for the constructs were anchored by 1 = "strongly disagree " to 7 = "strongly agree" **: Scale for this construct was anchored by 1="very poorly established" to 7= "very well established"*		

Table 1: Operationalization of Constructs

Modularization was operationalized as a second order construct, based on two formative dimensions, loose coupling and standardization. Project success was measured by meeting their functionality, cost and time constrains. Before continuing with the empirical analysis, we assessed the reliability and validity of the measurement model. Content validity is assumed as the items for the constructs were used in previous studies (see Table 1) and also through an initial pretest phase. We then validated our reflective measurements following the suggestions in the literature (Chin 1998). First, the items show internal consistency, with all Cronbach's alpha values greater than 0.7. Further, we checked for composite reliability, and find that all constructs exhibit values greater than 0.7. Additionally, the measurement item loadings on their constructs are greater than the required threshold. The average variance extracted (AVE) values are above the accepted limit of 0.5. Hence, the measurements exhibit convergent validity. The Fornell-Larcker criterion, according to which the AVE of each latent construct should be higher than the construct's highest squared correlation with any other latent constructs, (Hair et al. 2011), is also satisfied (Table 2).

	AVE	**C.A.**	**C.R.**	**OS**	**LC**	**ST**	**LTO**	**ECR**	**SUC**
OS	0.711	0.813	0.880	0.843*					
LC	0.795	0.744	0.886	-0.010	0.892*				
ST	0.862	0.840	0.926	-0.231	0.311	0.928*			
LTO	0.671	0.773	0.858	0.281	0.140	0.005	0.819*		
ECR	0.869	0.925	0.952	0.143	0.327	0.161	0.105	0.932*	
SUC	0.702	0.788	0.876	0.079	0.136	0.113	0.292	0.233	0.838*
*C.R.: Composite Reliability; C.A.: Cronbach's Alpha; AVE: Average Variance Extracted; * : Square root values of the AVE*									

Table 2: Measurement Model

The second order construct of IT modularity is measured by the first order constructs loose coupling and standardization.

Modularity:	**Loose Coupling**	**Standardization**
Indicators Weights	0.601**	0.633**
Variance Inflation Factor (VIF)	1.107	1.107
*Notes: *p < 0.05; **p < 0.01*		

Table 3: Formative Measures

These constructs have significant path coefficients (see Table 3). Overall, our measurement model indicates psychometric adequacy for further analysis. We also conducted a post-hoc power analysis to assess the sample size as suggested by Cohen (Cohen 1988). For the power analysis, we used the G*Power 3.1 Software (Faul et al. 2009). The post-hoc power analysis exhibited a power above the cut-off threshold of 0.8 (Cohen 1988) at the 95% confidence interval.

Empirical Analysis

We used structural equation modeling and the software SmartPLS 2.0.M3 (Ringle et al. 2005), to analyze our data. With SmartPLS no further sample distribution assumptions are necessary (Lohmöller 1989). The software was used to calculate path coefficients and to determine the paths' significance in the model using the bootstrapping function. The results of the analysis are shown in Figure 2. Overall, the structural model explains 21% of the variance of external component reuse and 5.4% of the variance of the software development project success. Different from our prediction in H_1, we find that organizational structure does not influence external software components reuse. However, after controlling for the size of the organization, we find that for large organizations, organic structure does positively influence external component reuse, partially supporting H_1. Different from what we predicted in H_2, LTO is not significant

in the model, which leads us to reject hypothesis H_2. Our results show that modularity is the most significant factor in explaining the use of external components, supporting hypothesis H_3. As theoretically predicted, we find that using external component positively influences software projects success, thus providing support for H_4. We also controlled for the size and age of the organization, but they were not significant in predicting the level of external component reuse.

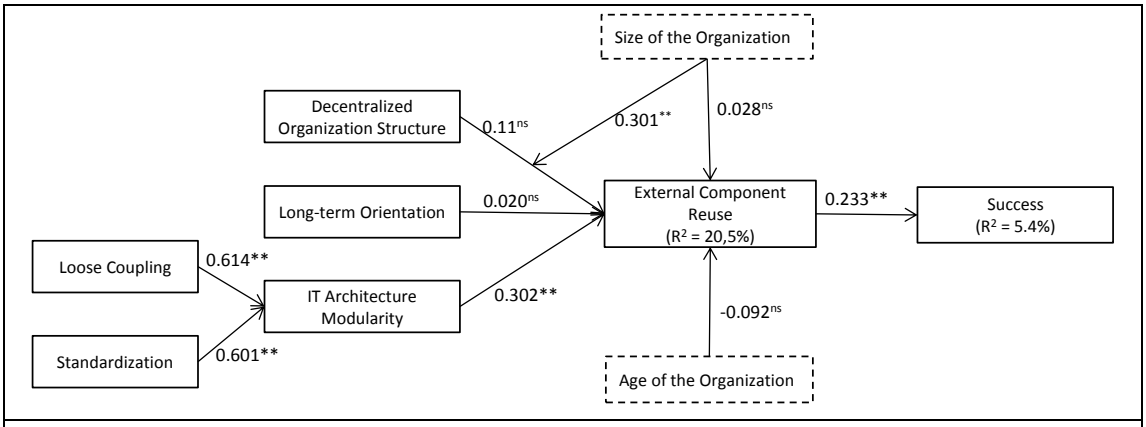

Figure 2: Empirical results for the first model

Conclusions, Discussion and Limitations

In this study we analyzed the reuse of external software components from a contingency perspective and we argue that every organization needs to choose the best strategy in order to create competitive advantage. Thus, we could provide some explanation for the external software reuse in organizations based on a set of contingency factors. We empirically found that the use of external software components leads to better outcomes for the software development projects.

We find that the structure of the organization plays a role only when it is moderated by the size. Thus, for large organizations, an organic structure would contribute to more external reuse. One could argue that small organizations, despite their fixed structures, tend to be fairly dynamic anyways. Whereas, this finding might seems trivial, it also confirms that structure of the organization and IT should be properly aligned, especially in disruptive settings. Also, we derived this hypothesis from innovation theory that mainly focuses on the perspective of large organizations, which may limit the applicability, as suggested in our data.

We also hypothesized that organizations, which have a LTO would prefer to not be dependent on external component vendors, and thus reuse less. This hypothesis was not supported by our data. However, an alternative explanation for this finding could be that some organizations actually view external reuse as a long-term strategy, because they can organize and deploy resources to achieve efficiency in the long-term. We find that organizations do use external software components in their projects, even after controlling for the age of the organization. Software reuse can thus be seen as an effective, novel organization strategy. In fact this insight is quite interesting for this study and need to be further explored in future research.

As expected, we also found that architecture modularity is the most significant driver when it comes to foster the reuse of software components, especially for small organizations. Thus, the higher the architecture modularity, the easier it is to integrate and exchange not only internal components, but also those procured from the market. This link between external software reuse and architecture modularity had not been empirically investigated. As Tiwana and Konsynski (2010) state, "architectural modularity also enhances IT alignment through mechanisms other than enhancing IT agility. For example, organizations with highly modular architectures might be able to decompose larger projects into incremental subprojects and more readily integrate off-the-shelf applications or modular open source

software components" (Tiwana and Konsynski 2010, p. 299). The actual expenditures on external software components for the majority of the companies in our sample is about 5% to 20% of the total budget. While this is still a considerable amount, it also means that a large amount of software is still developed in-house. Thus, modularity may increase choice and flexibility more when organizations can consider external versus internal component sources. Moreover, modularity fosters external knowledge integration, which enables organizations to further extend their boundaries.

From the business practice perspective, we recommend that software providers adopt modular software architectures using component-based reuse during their software production process. Therefore, adopting modular architectures can drive the exploitation of external knowledge through component reuse. Furthermore, we suggest that external sourcing is a viable alternative (and complement) to internal component sourcing models. Our study offers evidence that supports Shang et al. (2012) argument that a market-based solution would be the most efficient arrangement for external component sourcing. Additionally, since by adopting external sourcing strategies will further influences the ability of the team to recognize and quickly learn external technologies, it can further foster the performance of software projects. Hence, we argue that the industry should look into creating more efficient software component market places by using online exchange platforms. For large organizations, top management should also consider the structure of the organization, as it might have an effect on software development processes.

This study also has some limitations. First, the data is self-reported. Second, the results of the study are based on a fairly small data set and have not accounted for a specific industry type, which could provide further insights. Additionally, since we use cross sectional data, we can only show associations, not causality. Future research needs to account for the above limitation as well as to consider further contingency variables that could explain external reuse and its effects on software project success within organizations.

REFERENCES

Ayala, C., Hauge, Ø., Conradi, R., Franch, X., Li, J., and Velle, K. S. 2009. "Challenges of the Open Source Component Marketplace in the Industry," in *Open Source Ecosystems: Diverse Communities Interacting*. Springer, pp. 213-224.

Baldwin, C. Y., and Clark, K. B. 2006. "The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model?," *Management Science* (52:7), pp. 1116-1127.

Baumann, O. 2009. "Die Bedeutung Der Organisationstheorie Für Die Entwicklung Der Wirtschaftsinformatik," *Wirtschaftsinformatik* (51:1), pp. 72-81.

Benlian, A., and Hess, T. 2007. "A Contingency Model for the Allocation of Media Content in Publishing Companies," *Information & Management* (44:5), pp. 492-502.

Brereton, P., and Budgen, D. 2000. "Component-Based Systems: A Classification of Issues," *Computer* (33:11), pp. 54-62.

Burns, T. E., and Stalker, G. M. 1961. *The Management of Innovation*. London: Tavistock.

Buxmann, P., Diefenbach, H., and Hess, T. 2012. *The Software Industry: Economic Principles, Strategies, Perspectives*. Heidelberg: Springer.

Chin, W. W. 1998. "The Partial Least Squares Approach for Structural Equation Modeling," in *Modern Methods for Business Research,* G. Marcoulides (ed.). Mahwah, USA: Lawrence Erlbaum Associates, pp. 295-336.

Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*. Mahwah, USA: Lawrence Erlbaum Associates.

Covin, J. G., and Slevin, D. P. 1988. "The Influence of Organization Structure on the Utility of an Entrepreneurial Top Management Style," *Journal of Management Studies* (25:3), pp. 217-234.

Faul, F., Erdfelder, E., Buchner, A., and Lang, A.-G. 2009. "Statistical Power Analyses Using G* Power 3.1: Tests for Correlation and Regression Analyses," *Behavior Research Methods* (41:4), pp. 1149-1160.

Fixson, S. K., Ro, Y., and Liker, J. K. 2005. "Modularisation and Outsourcing: Who Drives Whom? A Study of Generational Sequences in the Us Automotive Cockpit Industry," *International Journal of Automotive Technology and Management* (5:2), pp. 166-183.

Frakes, W. B., and Isoda, S. 1994. "Success Factors of Systematic Reuse," *IEEE Software* (11:5), pp. 14-19.
Frakes, W. B., and Kang, K. 2005. "Software Reuse Research: Status and Future," *IEEE Transactions on Software Engineering* (31:7), pp. 529-536.
Gawer, A., and Cusumano, M. A. 2014. "Industry Platforms and Ecosystem Innovation," *Journal of Product Innovation Management* (31:3), pp. 417-433.
Hair, J. F., Ringle, C. M., and Sarstedt, M. 2011. "Pls-Sem: Indeed a Silver Bullet," *The Journal of Marketing Theory and Practice* (19:2), pp. 139-152.
Herrmann, A., Gassmann, O., and Eisert, U. 2007. "An Empirical Study of the Antecedents for Radical Product Innovations and Capabilities for Transformation," *Journal of Engineering and Technology Management* (24:1–2), pp. 92-120.
Laverty, K. J. 1996. "Economic "Short-Termism": The Debate, the Unresolved Issues, and the Implications for Management Practice and Research," *The Academy of Management Review* (21:3), pp. 825-860.
Lawrence, P. R., and Lorsch, J. W. 1986. *Organization and Environment: Managing Differentiation and Integration*. Boston, USA: Harvard Business Press.
Loewenstein, G., and Thaler, R. H. 1989. "Anomalies: Intertemporal Choice," *The Journal of Economic Perspectives* (3:4), pp. 181-193.
Lohmöller, J.-B. 1989. *Latent Variable Path Modeling with Partial Least Squares*. Heidelberg, Germany: Physica-Verlag.
McIlroy, D. 1969. "Mass-Produced Software Components," *Proceedings of Software Engineering Concepts and Techniques,* J.M. Buxton, P. Naur and B. Randell (eds.), Garmisch, Germany, pp. 138-155.
Nidumolu, S. R. 1996. "Standardization, Requirements Uncertainty and Software Project Performance," *Information & Management* (31:3), pp. 135-150.
Pavlou, P. A., and El Sawy, O. A. 2006. "From It Leveraging Competence to Competitive Advantage in Turbulent Environments: The Case of New Product Development," *Information Systems Research* (17:3), pp. 198-227.
Ravichandran, T., and Rothenberger, M. A. 2003. "Software Reuse Strategies and Component Markets," *Communications of the ACM* (46:8), pp. 109-114.
Ringle, C. M., Wende, S., and Will, A. 2005. "Smartpls 2.0.M3." Retrieved 07.2015, from http://www.smartpls.de
Shang, R. D., Mohan, K., Lang, K. R., and Vragov, R. 2012. "A Market Mechanism for Software Component Reuse: Opportunities and Barriers," in: *Proceedings of the 14th Annual International Conference on Electronic Commerce*. Singapore, Singapore: ACM, pp. 62-69.
Sherif, K., Appan, R., and Lin, Z. 2006. "Resources and Incentives for the Adoption of Systematic Software Reuse," *International Journal of Information Management* (26:1), pp. 70-80.
Sojer, M., and Henkel, J. 2010. "Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments," *Journal of the Association for Information Systems* (11:12), pp. 868-901.
Stefi, A., and Hess, T. 2015. "To Develop or to Reuse? Two Perspectives on External Reuse in Software Projects," in *Software Business,* J.M. Fernandes, R.J. Machado and K. Wnuk (eds.). Springer International Publishing, pp. 192 -206.
Tiwana, A. 2008. "Does Technological Modularity Substitute for Control? A Study of Alliance Performance in Software Outsourcing," *Strategic Management Journal* (29:7), pp. 769-780.
Tiwana, A., and Konsynski, B. 2010. "Complementarities between Organizational It Architecture and Governance Structure," *Information Systems Research* (21:2), pp. 288-304.
Verwaal, E., Commandeur, H., and Verbeke, W. 2008. "Value Creation and Value Claiming in Strategic Outsourcing Decisions: A Resource Contingency Perspective," *Journal of Management* (35:2), pp. 420-444.
Wang, T., and Bansal, P. 2012. "Social Responsibility in New Ventures: Profiting from a Long-Term Orientation," *Strategic Management Journal* (33:10), pp. 1135-1153.
Weill, P., and Olson, M. H. 1989. "An Assessment of the Contingency Theory of Management Information Systems," *Journal of Management Information Systems* (6:1), pp. 59-85.
Wong, C. W. Y., Lai, K.-h., and Cheng, T. C. E. 2011. "Value of Information Integration to Supply Chain Management: Roles of Internal and External Contingencies," *Journal of Management Information Systems* (28:3), pp. 161-200.
Woodward, J. 1958. *Management and Technology*. London: Her Majesty's Stationary Office.